Appendix E

Title:
Macro Specification
DMUT Version 2.2

1 Introduction

1.1 Overview

The Transmit Data Management Unit (DMUT) provides direct data transfer from the shared memory to the internal Transmit Buffer (TB) with minimal host CPU intervention. The data buffers located in the shared memory are associated with one of the logical channels.

The on-chip Transmit Buffer (TB) requests data from the DMUT and after receiving it sends them to a Transmit Protocol Machine for protocol (HDLC, Ethernet, Frame Relay, ATM,...) processing.

A linked list of descriptors associated to each channel is located in the shared RAM and handled by the DMUT. The address generator of the DMUT supports full link list handling. The descriptors can be stored in separate memory location from the data buffers themselves allowing full scatter/gather assembly of packets. In order to have only one read access on the system bus for each descriptor, the current descriptor of each channel is hold on-chip. The interrupt vectors generated by the DMUT are transferred to a separate interrupt controller (DMUI). The DMUI will transfer the interrupt vectors of the complete device to the shared memory.

The number of channels is a flexible parameter of the DMUT macro. The DMUT can be used for multichannel devices (M256) as well as for high speed controllers with a reduced number of channels (DSCC4).

1.2 Features

- Data transmit from the shared memory to the central Transmit Buffer (TB)
- Support of linked list data structures located in the shared memory, consisting of a descriptor associated to a data section
- Independent channel configuration
- Forward interrupt vectors to the DMUI

1.3 System Integration

The DMUT has four interfaces:

- FPI Initiator Bus
- FPI Target Bus
- TB interface
- DMUI interface

The data are read from the FPI initiator bus and written to the TB interface. The descriptors are also accessed through the FPI initiator bus. The interrupt vectors are transferred on the DMUI interface bus. Each channel is configured independently (base address register, command register,...) via the FPI target bus.

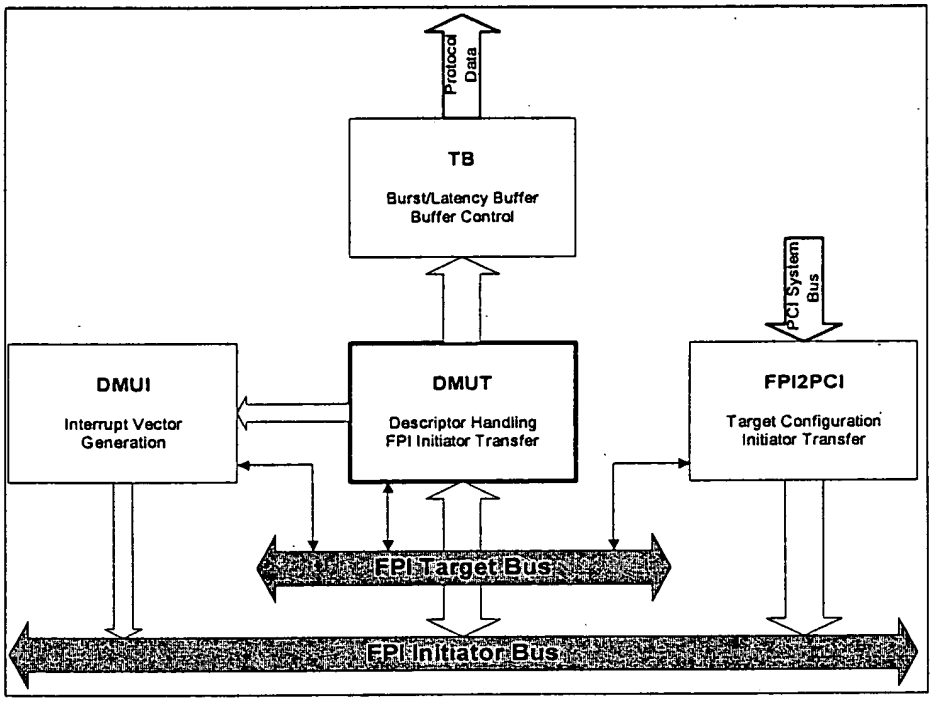


Figure 1
System Integration

1.4 Known Restrictions and Problems

The priorization on the FPI initiator is not an easy task.

DMUT should have priority if new data or a descriptor change is required in the middle of a frame in order to prevent underrun of the transmit buffer (and an abort of the frame). But at the beginning of a frame, the transfer is not time critical.

DMUT and DMUI should have an equal priority on the FPI initiator bus to prevent buffer underrun in the transmit buffer and buffer overflow in the IC.

2 Functional Description

2.1 Block Diagram incl. Clocking Regions

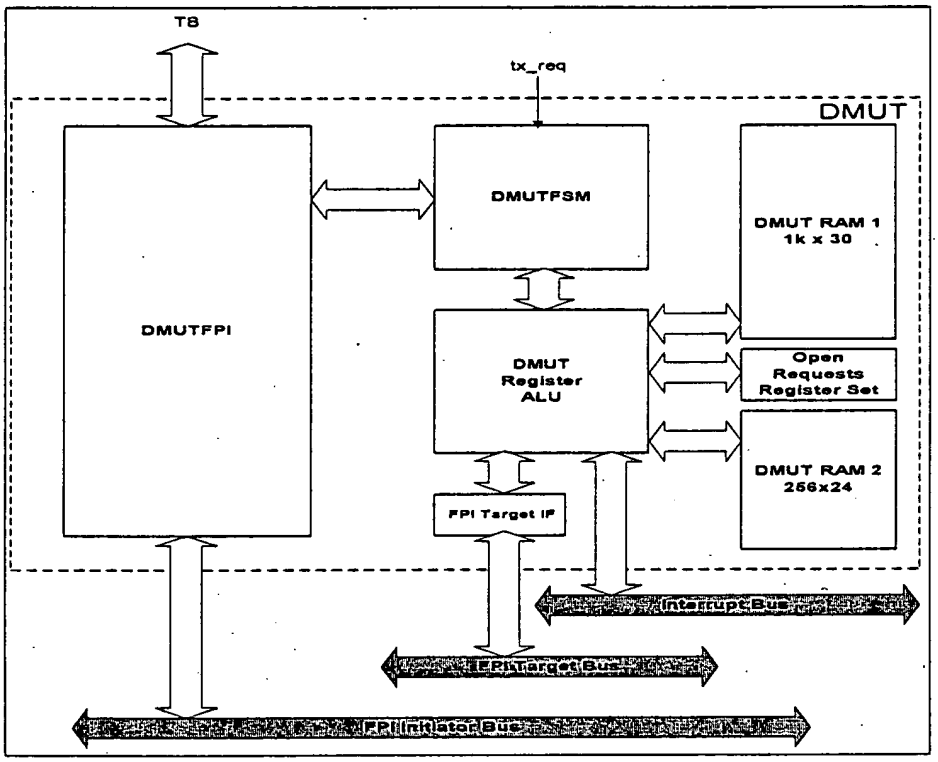


Figure 2
DMUT Block Diagram

The DMUT macro consists of 4 blocks, an open requests register set and 2 DMUTRAMs, that store the current transmit descriptor, the next transmit descriptor, the state of the channel, the interrupt mask and the interrupt queue number. The DMUTFPI contains a FPI initiator interface and is directly connected to the FPI Initiator Bus on one side and to the transmit buffer (TB) on the other side. This approach permits a fast transmission from the initiator bus to the transmit buffer without intermediate storage. The DMUTFSM handles the requests of the TB, controls the address and burst length calculation, initiates the data transfers from the shared memory to the TB and the updating of the current transmit descriptors by triggering the DMUTFPI. This block also controls the assembling of the interrupt vectors and writes them to the register block. There they will

be stored until the request on the interrupt bus was granted and the interrupt information could be transmitted. The register block also stores information of the current channel available for the DMUTFSM and the ALU calculates addresses and burst lengths based on requested amount from the TB and the stored information in the DMUTRAM. The requests which could not have been responded so far are stored in the open requests register set. Configurational access from the CPU to the DMUT macro is possible via FPI Target Bus and FPI Target interface (refer to "SMIF Specification").

The complete DMUT block and all the interfaces are synchronous to the same CLK signal derived from the FPI initiator bus.

2.2 Normal Operation Description

The DMUT operates with a linked list of descriptors pointing to a data section. For each active channel a linked list is allocated in the external shared memory by the CPU. The current descriptor contains the pointer to the next descriptor, the start address of the data section and the reserved size of the data section.

For initialization the CPU programs the First Transmit Descriptor Address (FTDA) register, the interrupt mask, the interrupt queue and the Virtual Channel Specification command register (CSPEC_CMD) with the transmit init command(refer to chapter chapter 4.2). The byte swap mode (little/big endian - common for all channels) and the maximum burst length on the IFPI bus (also common for all channels) are programmend in global registers. The DMUT then fetches immediately all this information after an transmit init command and stores them in the chip internal RAM. When the transmit buffer first requests data for this channel the entire descriptor pointed by FTDA from the shared memory will be read. After storing the entire descriptor information in the on chip RAM, the DMUT will generate a Command Complete Interrupt (CMDC).

Table 1
Request Register (read access)

Bit	31	30 .. RLB+16	RLB+15... 16	15 . . CNB	CNB-1... 0
Function	DEL	Reserved	Requested Transfer Length RTL	Reserved	Channel number (CHN)

Table 2
Request Register (write access)

Bit	31	30 . . . RLB+16	RLB+15 . . . 16	15..CNB	CNB-1 . . . 0
Function	CI	Reserved	Served Transfer Length STL	Reserved	Channel number (CHN)

As long as the amount of empty locations in the transmit buffer is above the programmable threshold for a particular channel, TB will request a DMUT transfer. The DMUT will then read the request register at the TB-DMUT interface consisting of the DEL Bit (if set), the channel number (CHN) and the number of words (RTL) the TB requests from DMUT.

The DMUT reads the Transmit Descriptor corresponding to the channel number CHN out of its on-chip RAM and calculates the maximum number of bytes that can be read from the current transmit data section in the external memory based on checking the Byte Number (NO) field (part of the transmit descriptor), the amount of open request from former transfers, the requested transfer length and the maximum burst length.

The data transmission to the transmit buffer starts with the write of the Request Register consisting of the channel number and the served transfer length of the current transfer. Note that this amount of data might be smaller after initialization due to an end of frame (specific Frame End (FE) bit in the transmit descriptor) than the Requested Transfer length (RTL) . If this transfer contains a Frame End or a Transmit Abort Statusword the DMUT sets the Complete Indication bit (CI). During the transfer of a statusword the side-band signal dt_tx_stat is active. The transfer will be initiated by the DMUTFSM and executed by the DMUTFPI.

Depending on NO in the current transmit descriptor, the open requests and RTL one or several read accesses must be performed by the DMUT on the FPI initiator bus. The DMUT stops serving this request as soon as the requested amount of data was transfered to the TB, the DMUT transferred the maximum burst length (scalable in global register CONF3), when a Frame End bit (FE) in the current transmit descriptor is set or when the channel was aborted using an Transmit abort command (currently ongoing transfers will not be interrupted by this command). The difference between RTL and STL, called the open requests, which were not served in this cycle, will be stored in the open requests register set separately for each channel. When the TB requests new data for this channel CHN, the DMUT calculates the served transfer length STL out of the open requests, RTL, NO in the current transmit descriptor and the maximum burst length. Therefore Served Transfer Length might be more, equal or less than the Requested Transfer Length (RTL), but the maximum served length for one transfer cycle is 65

DWORDS (scalable in CONF3 to smaller maximum values for different system requirements) in the M256F application (max. for the FPI2PCI bridge - assuming that the data section contains enough data). The maximum amount of data the transmit buffer might request is 8K bytes.

In order to treat open request and requests from the transmit buffer equally, there is a continous switch between these 2 kinds of requests, if both are existent at the time. In case of the open requests, the DMUT scans through its open request register set and looks for open request which have not been served so far. If there are open requests for a channel, data transmission will be initiated. The procedure is the same as described above.

The DMUT will not branch to the next Transmit Descriptor of a channel when a FE Bit is set and all data were transferred, it serves a new request from the TB for another channel or looks for open requests in its register set. So open requests from other channels can get served faster, possible underruns might be avoided. The next Transmit Descriptor will be retrieved with the next data transfer of the channel.

During data transmission a channel might be reconfigured in order to handle more or less data. Therefore the reserved buffer size per channel in the transmit buffer has to be modified. The CPU turns this particular channel off using the transmit off command. The transmit buffer will free the allocated locations in the buffer for this channel and requests the DMUT to delete all stored open locations in the onchip RAM for the channel (DEL bit set in the request register). If this channel is just served by DMUT, data will be transferred to the transmit buffer, data have to be discarded there.

The DMUT will be informed about the tranmsit off command with the DEL bit in the request register set, the requested transfer length is invalid. The DEL bit will only be evaluated by the DMUT during reading the request register. The DMUT clears the open locations, sets the C-Bit in the Transmit Descriptor (if necessary) and generates a Command Completed Interrupt (CMDC) and a HI Interrupt (if HI bit is set in the Transmit Descriptor). Afterwards no more requests for this channel will be generated by the transmit buffer. After the CMDC Interrupt the CPU can set up the channel again (via INIT command) using another buffer size for the transmit buffer.

The DMUTFPI will perform split block read transfer (refer to FPI specification) to the FPI2PCI bridge. It writes data directly to the TB interface with minimum storage when the data are valid on the FPI intiator bus. If TB inserts wait states, DMUT can also insert wait states on the FPI initiator bus for the split response. Due to the latency on the PCI bus it may not be assumed that a requested data packet will be transfered in one cycle by the PCIFPI bridge, several smaller packets might be possible.

If the DMUT completes reading the data section associated with the Tx Descriptor, then the DMUT will set the Complete (C) bit in the Tx Descriptor if the CEN (Complete Enable) bit is set (refer to chapter chapter 2.3). Additionally an HI interrupt vector will be transferred to the DMUI if the HI bit is set in the Tx Descriptor. The DMUT will then branch to the next transmit descriptor. When a frame end is detected and all data have

been transferred to the TB, the DMUT will add a frame end status word with/without data in order to inform the protocoll machine about the frame end.

The data transfer is controlled by the HOLD bit (CPU is owner of it), which is located in the transmit descriptor. If the HOLD bit is not set the DMUT will then branch off to the next descriptor. If the HOLD bit is set in the current Tx Descriptor this one is the last descriptor of the linked list available for the DMUT. The corresponding DMUT channel is deactivated as long as the CPU does not request an activation via the Virtual Channel Specification command register (refer to chapter chapter 4.2).

If the HOLD bit is detected in a descriptor and the Frame End bit is not set the DMUT will transfer all data of the belonging datasection. Afterwards the DMUT will set the C bit (if CEN is set) and generate an Hold Caused Transmit Abort Interrupt (HTAB) interrupt (the HI bit will be set in the interrupt vector as well when the HI bit is set in current transmit descriptor) in order to inform the CPU about the erroneous descriptor structure and writes an abort status word to the transmit buffer (protocoll machine info), since a complete frame could not be transmitted.

Then the DMUT reads the Current Transmit Descriptor once more. If the Hold Bit has been removed it will branch off to the Next Transmit Descriptor. When the Hold Bit is still set an internal Poll Bit will be set. Any further requests from the TB for this channel will not be responded, the number of requested data from the TB will be written in the open request register. In the meanwhile the CPU may issue a Transmit Abort Command for this channel. In this case, the DMUT will not repoll the old Transmit Descriptor and just issue a CMDC interrupt, since an abort indication was already sent to the protocoll machine. The DMUT continues with the descriptor pointed by FTDA.

If the HOLD bit and the Frame End Bit are set in the descriptor, the DMUT transfers the data of the belonging data section to the transmit buffer. At the end of the datasection it appends a frame end status word (trigger for protocoll machine). If the CEN bit is set, additionally the C-bit will be set in the current transmit descriptor, if the HI bit is set in the current transmit descriptor an HI Intr will be generated. When a new request for this channel appears (from tb or from the open request registers) the DMUT repolls the FE/ HOLD descriptor. If the HOLD bit has been removed it will branch off to the next transmit descriptor. If the Hold bit is still set, the internal poll bit will be set and no HTAB INTR will be generated. Nor data neither a statusword will be sent to the transmit buffer in this case. Any further requests from the TB for this channel will not be responded, the number of requested data from the TB will be written in the open request registers.

Note that after a request from TB for a channel, which is on hold, the request register in the TB will not be written, since no data transfer will take place. Channels on HOLD will not be visible during scaning the open requests register set. When new data in the shared memory is available, the host CPU performs a Transmit Hold Reset command, the internal Poll Bit will be reseted.The status of this channel will be switched to ACTIVE in the onchip RAM, requests from the TB will be answered and open locations will be visible in the onchip open register set. In both cases the DMUT repolls the old HOLD

Descriptor, the HOLD Bit is suposed to be removed and it branches to the Next Transmit Descriptor. If the HOLD bit is not removed (erroneous programming by CPU) no action will take place. The transmit abort command is another way to switch a channel from HOLD to ACTIVE. In this case, the open requests will be made visible but the old descriptor will not be repolled and data transmission starts with the transmit descriptor pointed by FTDA .

The CPU will always issue a transmit hold reset command when it removes the Hold bit in a descriptor, no matter the DMUT has already seen the Hold bit or not.

The status of each channel (active or hold) is stored in the onchip RAM of the DMUT.

Although the DMUT works only 32-bit word oriented, it is possible to begin a transmit data section at an uneven address. The two least significant bits (ADD) of the transmit data pointer determine the beginning of the data section and the number of bytes in the first 32-bit word of the data section, respectively.

Table 3 shows how many bytes are valid in the first 32-bit word depending on the ADD bits for both little endian and big endian mode. Of course, the number of valid bytes depends on the length NO of the data buffer:

Table 3
Meaning of ADD in Little/Big Endian Mode

ADD	Number of valid bytes	Little Endian				Big Endian			
00	4, if NO > 3	byte 3	byte 2	Byte 1	byte 0	byte 0	byte 1	byte 2	byte 3
01	3, if NO > 2	byte 2	byte 1	Byte 0	-	-	byte 0	byte 1	byte 2
10	2, if NO > 1	Byte 1	byte 0	-	-	-	-	byte 0	byte 1
11	1, if NO > 0	byte 0	-	-	-	-	-	-	byte 0

If the big endian byte ordering is programmed, the DMUT provides byte ordering swapping.

Note: The descriptor transfer is always a 32 bit access (no byte swapping).

2.3 Transmit Descriptor Definition

The transmit descriptor is initialized by the host CPU in the shared memory and is read by the DMUT. The address pointer to the first transmit descriptor will be stored in the onchip RAM, when requested to do so by the host via Transmit Init Command. The transmit descriptor is read entirely when the transmit buffer requests a data transfer for this channel and is stored in the on-chip memory. It will also be read entirely when branching from one transmit descriptor to the next. Therefore all information in the next descriptor must be valid when the DMUT branches to this descriptor.

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Byte Address																
0x00	FE	HOLD	HI	CEN	Reserved						Descriptor ID					
0x04	Next Transmit Descriptor Pointer															
0x08	Transmit Data Pointer															
0x0C	Rsvd	C	Reserved													

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Byte Address																
0x00	NO															
0x04	Next Transmit Descriptor Pointer															
0x08	Transmit Data Pointer															
0x0C	Reserved															

FE: Frame End, set by the host

It indicates that the current transmit data section (addressed by Transmit Data Pointer) contains the end of a frame. The DMUT will transfer data from this section to the transmit buffer (TB) and it checks the HOLD bit value. When HOLD=0 it branches to the next Tx Descriptor with the next request for this channel.

HOLD: Hold

It indicates whether the current descriptor is the last element of a linked list or not:

- HOLD=0: A next descriptor is available in the shared memory; after checking the HOLD bit stored in the on-chip memory the DMUT branches to next transmit descriptor.
- HOLD=1: The current descriptor is the last one that is available for the DMUT. The corresponding DMUT channel is deactivated for transmit direction as long as the CPU does not request an activation via the command register.

NO: Byte Number

The byte number defines the number of bytes stored in the data section to be transmitted. Thus the maximum length of data buffer is 65535 bytes. In order to provide dummy transmit descriptors NO = 0 is allowed in conjunction with the FE bit set. In this case (NO = 0) a HI INTR and/or the C-bit will be generated/set when the DMUT recognizes this condition. If NO = 0 without FE set two different cases are possible:

a) The last Transmit Descriptor contained a Frame End. Since no data were transfered for this transmit descriptor and the last frame was closed, no abort status word and no abort INTR will be generated.

If the hold bit is set (HOLD = '1'), a HTAB Intr will be issued in order to inform the CPU about the erroneous descriptor structure and the current Transmit Descriptor will be repolled immediately. If the HOLD bit is still set, the internal poll bit will be set. In order to move on, a Transmit Hold Reset (THR), a Transmit Abort or a Transmit Off Command can be issued. If the HOLD bit is removed after the repoll, the DMUT will branch to the next Transmit Descriptor.

If the HOLD bit is not set, the DMUT will branch to the next Transmit Descriptor

b) The last Transmit Descriptor did not contain a Frame End. An Abort Statusword will be generated and the corresponding TAB INTR issued.

If the HOLD Bit is not set in the current Transmit Descriptor, the DMUT branches off to the next transmit descriptor.

If the HOLD Bit is set, a HTAB Intr will be issued and this Transmit Descriptor will be repolled immediately. If the HOLD Bit is still set, the internal poll bit will be set. In order to move on, a Transmit Hold Reset (THR), a Transmit Abort or a Transmit Off Command can be issued. If the HOLD bit is removed after the repoll, the DMUT will branch to the next Transmit Descriptor.

The data bytes must be stored within the transmit data section according to the selected mode (little endian or big endian).

HI: Host Initiated Interrupt

If the HI bit is set, the DMUT transfers an HI interrupt vector to the interrupt controller after transferring all data bytes of the current data section to the internal transmit buffer.

Next Transmit Descriptor Pointer

This 32-bit pointer contains the start address of the next transmit descriptor, it has to be dword aligned. After sending the indicated number of data bytes, the DMUT branches to the next Tx Descriptor to continue transmission. The Tx Descriptor is read entirely at the beginning of transmission and stored in on-chip memory. Therefore all information in the next descriptor must be valid when the DMUT branches to this descriptor.

This pointer is not used if a Transmit Abort command for this channel is detected while the DMUT still reads data from the current transmit descriptor. In this case the First Transmit Descriptor Address (FTDA) register is used as a pointer for the next transmit descriptor to be branched to.

Transmit Data Pointer

This 32-bit pointer contains the start address of the transmit data section. Although the DMUT works long word oriented, it is possible to begin transmit data section at byte addresses.

C: Complete

This bit is set by the DMUT if

- it completes reading data section normally
- it was aborted by a transmit off or transmit abort command.

The Complete bit releases the descriptor (owner bit).

CEN: Complete Enable

This bit is set by the CPU if the complete bit mechanism is desired:

- CEN=0: the DMUT will NOT update the Tx Descriptor with the C bit. In this mode the use of the HI interrupt is recommended.
- CEN=1: the DMUT will set the C bit.

Descriptor ID

This field is read by the DMUT and written back in the corresponding interrupt status vector for HI , TAB or HTAB interrupt. This value provides a link between the interrupt vector and the Tx Descriptor, to be used by the software.

2.4 Interrupt Vector Definition

The DMUT generates 2 groups of interrupts, channel command related interrupts and interrupts which are related to the data transfer of the DMUT for a particular channel. These 2 groups will be distinguished by the Interrupt ID. Additionally the channel number and specific bits are part of the interrupt vector. The command interrupts will be written to a designated queue with a fixed length. Channel interrupts can be written to 8 different queues with variable length, the specific queue for each channel will be programmed during initializiation of the DMUT.

For Channel Interrupts the Descriptor ID value is written back to the corresponding interrupt vector. This provides a link between the interrupt vector and the descriptor, to be used by the software. The interrupt vectors are transferred to the interrupt controller and have following format:

Table 4
Interrupt Vector Format (Channel Interrupt Vector)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
1	0	1	1	1	Q2	Q1	Q0	0	0	Descriptor ID					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HI	TAB	0	HTAB	0	0	0	0	Channel No.							

Command Interrupt Vector

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	CMDC

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	Channel No.							

Channel number

Identifies the channel where the interrupt occurred

Descriptor ID

Identifies the transmit descriptor, which generated the interrupt

HI: Host Initiated Interrupt

If the HI bit in the transmit descriptor is set, an Interrupt Vector with this bit set is generated when DMUT finishes the old transmit descriptor and branches to the next transmit descriptor.

Q2,Q1,Q0: Queuenumber

Identifies the number of the interrupt queue for the channel in the shared memory (needed by the interrupt controller - not valid for the Command Intr.)

TAB: Transmit Abort

Issued if the DMUT detects a transmit abort command or an erroneous transmit descriptor (FE=0 und NO=0) and a frame was not transmitted completely for this channel.

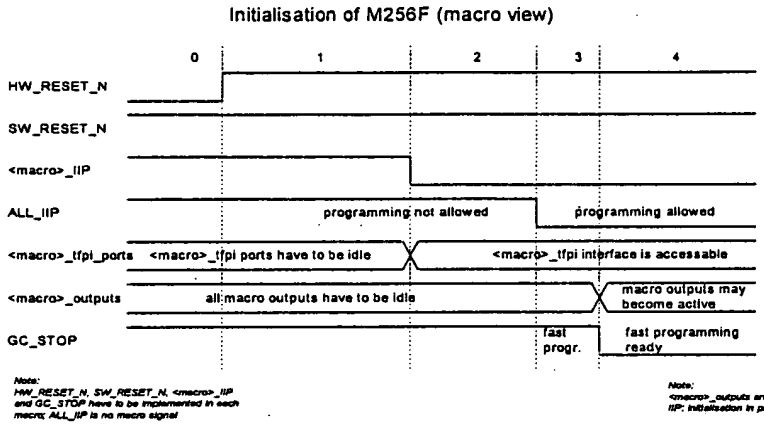
HTAB: Hold Caused Transmit Abort

Issued if the DMUT retrieves a transmit descriptor where HOLD =1 and FE = 0. The interrupt will be generated after the data section was transferred completely.

CMDC: Command Completed

CMDC = 1 identifies a successful channel command performed by the host CPU. The DMUT is ready for a new command by the CPU for this channel.

2.5 Reset Behavior



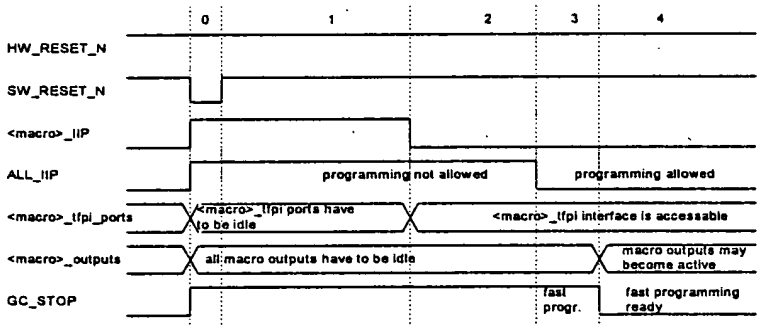
0: HW_RESET_N is active and deactivates SW_RESET_N; all macros are reset; GC_STOP set to '1'

1: HW_RESET_N inactive => each macro executes the macro specific initialisation of rams

2: some macros are ready with initialisation of rams, some not (ALL_IIP is a or-function of all <macro>_IIPs). The "ready macros" could be accessed by TFPI interface, but software programming is not allowed as long as ALL_IIP is active

3: all macros are ready with initialisation => ALL_IIP becomes inactive; software polls the ALL_IIP bit after HW_RESET_N; if ALL_IIP = 0 then software programms all macros (in a default "fast programming mode", GC_STOP = 1; reset value of GC_STOP is '1', but can be reset at any time)

4: after fast programming macro outputs (non TFPI) may become active and macro has to react on non FPI inputs



0: SW_RESET_N becomes active; all macros (registers) are reset; GC_STOP is active

1: SW_RESET_N inactive => each macro executes the macro specific initialisation of rams

2: some macros are ready with initialisation of rams, some not (ALL_IIP is a or-function of all <macro>_IIPs). The "ready macros" could be accessed by TFPI interface, but software programming is not allowed as long as ALL_IIP is active

3: all macros are ready with initialisation => ALL_IIP becomes inactive; software polls the ALL_IIP bit after SW_RESET_N; if ALL_IIP = 0 then software programms all macros in a "fast programming mode", GC_STOP = 1

4: after fast programming macro outputs (non TFPI) may become active and each macro has to react on non FPI inputs

All internal registers and functions are initialized to known states (RESET state).

In addition to this global reset, each channel can be turned off or aborted via the command register (refer to chapter 4.2.1).

2.6 Functional Test Description

2.7 Production Test Description

3 Macro Interfaces and Signal Description

All signals are active high until otherwise specified. Active low signals are designated by "_N" appended to their names. To make the design as re-usable as possible, a bus signal whose width is application dependent is specified with one of the following parameters:

Table 5
Application specific Parameters

Parameter name	Bus Type	Typical value (bits)
CNB	Channel Number Bus	8
DBB	Data/Status Bus	32
BLB	Burst Length Bus	6
RLB	Request Length Bus	13

3.1 Signal Description

In the following sections, "Flexible Peripheral Interconnect (FPI) Bus compliant" means that the specified bus uses a subset of the FPI features and satisfies the basic address and data cycle. Not all FPI signals are implemented because default values are sufficient for the application i.e. they can be coded as constants in the hardware. Refer to the FPI bus specification for details of the complete bus.

The following tables list the FPI-Bus signals and two additional out-band signals:

- dt_tx_stat to indicate if transferred word is status instead of pure data. This signal is only active during the transfer of a statusword (1 clock cycle).
- tx_req_n to indicate that the Transmit Buffer Action Queue (TBAQ) in the TB is not empty.

Table 6
Macro Interfaces and Signal Description

Symbol name	I/O	Function
Control Signals		
sysclk	I	system clock
reset_n	I	Reset
scanmode	I	Scanmode
gc_stop	I	global stop signal, used for fast RAM programming by the CPU
dt_iip	O	dmut macro initialization in progress, when active the DMUT macro initializes its internal RAMs
Transmit Buffer (TB) interface		
dt_tx_rd_n	O	Read Control (for request register)
dt_tx_wr_n	O	Write Control
dt_tx_a	O	1 bit address bus 0 => status port. 1 => data port.
dt_tx_d[DBB-1:0]	O	Output data/status from DMUT to TB controller
tx_d[DBB-1:0]	I	Input data from TB (request register)
tx_rdy	I	End of data transfer indication. 0 => DMUT should insert wait state. 1 => transmit buffer will finish transfer during this clock cycle.
dt_tx_stat	O	Current transferred word is statusword
tx_req_n	I	Service request from transmit buffer to DMUT controller

Table 6
Macro Interfaces and Signal Description (cont'd)

Symbol name	I/O	Function
Interrupt Controller (DMUI) Interface		
dt_int_req_n (**)	O	Interrupt Bus Request Line
ic_dt_gnt_n(**)	I	Interrupt Bus Grant Line
dt_int_d[DBB-1:0]	O	Interrupt Vector
dt_int_d_en[7:0]	O	Output Enable Bus for data
FPI Target Interface		
tfpi_a[8:2]	I	Address bus.
tfpi_d[DBB-1:0]	I	Data bus. Active during data phase.
tfpi_rdy	I	Ready Input, other target will finish datat cycle in the clock cycle (tfpi_rdy_n = 1) or it will insert wait states
tfpi_wr_n tfpi_rd_n	I I	Read/Write control. Following codes are defined: WR_N = 1; RD_N = 1 => NOP WR_N = 0; RD_N = 1 => data written to DMUT WR_N = 1; RD_N = 0 => data read from DMUT
tfpi_vc_sel_n	I	DMUT Slave select (registers of virtual channel specification) 0 => address is valid 1 => address is not valid
tfpi_vg_sel_n	I	DMUT Slave select (global registers) 0 => address is valid 1 => address is not valid
dt_tfpi_d[DBB-1:0]	O	Data bus output. Active during data phase.
dt_tfpi_d_en[7:0}	O	Data bus output enable bus (one enable line drives 4 data lines)
dt_tfpi_rdy	O	End of transfer indicator: 0 => Master should insert wait states 1 => DMUT will complete transfer in this cycle
dt_tfpi_rdy_en	O	Output Enable signal for Ready Output

Table 6
Macro Interfaces and Signal Description (cont'd)

Symbol name	I/O	Function
FPI Initiator Bus		
ifpi_gnt_n	I	DMUT Initiator Grant Line
ifpi_opc[3:0]	I	Operation Code (Input for split read response)
ifpi_d[DBB-1:0]	I	Data Bus
ifpi_sel_n	I	DMUT select signal, used for split block response
ifpi_ack[1:0]	I	Slave Response code
ifpi_rdy	I	Slave (PB) on data bus will be able to finish transaction in current clock cycle
dt_ifpi_req_n	O	DMUT Initiator bus request line
dt_ifpi_rd_n	O	Read Control Output
dt_ifpi_rd_n_en	O	Read Control Output Enable
dt_ifpi_wr_n	O	Write Control
dt_ifpi_wr_n_en	O	Write Control Output Enable
dt_ifpi_abort_n	O	Abort signal
dt_ifpi_abort_n_en	O	Abort Output Enable
dt_ifpi_opc[3:0]	O	Operation Code
dt_ifpi_opc_en	O	Operation Code Output Enable
dt_ifpi_tc[5:0]	O	Transfer Count (max. burst size 64dwords)
dt_ifpi_tc_en[1:0]	O	Transfer Count Output Enable Bus (one enable line drives 4 data lines)
dt_ifpi_a[DBB-1:0]	O	Address Bus
dt_ifpi_a_en[7:0]	O	Address Output Enable Bus
dt_ifpi_d[DBB-1:0]	O	Data Bus
dt_ifpi_d_en[7:0]	O	Data Bus Output Enable Bus
dt_ifpi_tag[3:0]	O	Tag Bus (used to transfer the master ID for Split Block Transfers)
dt_ifpi_tag_en	O	Tag Bus Output Enable

Table 6
Macro Interfaces and Signal Description (cont'd)

Symbol name	I/O	Function
dt_ifpi_rdy	O	DMUT will be able to finish transaction in the current clock cycle
dt_ifpi_rdy_en	O	Ready Output Enable

(**) These signals connect DMUI and the arbitration logic on the DMUI bus.

3.2 Data Flow and Functional Timing

3.2.1 FPI Master Bus

Refer to the SIEMENS FPI Bus for Munich-Macros Specification

3.2.2 FPI Slave Bus

Refer to the SIEMENS FPI Bus for Munich-Macros Specification

3.2.3 Interface between DMUT and Transmit Buffer

The DMUT interface to the TB is based on a bidirectional FPI like bus. Because only one operation code is defined (Single Word Transfer), an OPC signal is not necessary.

The tranmsit buffer requests data and the DMUT either serves the request (requested length < served length, requested length = served length, requested length > served length) or not (channel on HOLD). Before the DMUT starts the data transfer to the TB it writes the amount of data, the channel number and the CI (if necessary) to the request register (refer to chapter 2.2).

The select signal SEL_N is implicitly active and not physically present. DMUT sees TB as a request register at address 0 and a FIFO data port at address 1. Only 1 address bit (dt_tx_a) is defined on the bus.

Two out-of-band signals are required

1. tx_req_n to indicate that TB FIFO has empty locations.

2. dt_tx_stat to indicate the currently transmitted word is status word instead of pure data.

When the signal tx_req_n is active and the DMUT is in idle state, the DMUT initiates an address cycle by asserting address 0 (request register). During the data cycle, TB returns the request register, which describes the port status and asserts tx_rdy.

If the channel is not on HOLD, the DMUTFSM calculates the Served Transfer Length and writes STL together with the channel number and the CI bit (if necessary) into the request register (address 0). The DMUTFSM initiates the transfer writing the start address and the amount of data to the DMUTFPI. The DMUTFPI block requests the data via FPI Initiator bus from the shared memory and then transfers STL words with either individual or overlapped cycles. In best case, a burst cycle can occur without wait states but internally, the PMT interface of the transmit buffer has higher priority. In the following figure, a complete transfer (read request register, write request register, write data) with collision with the PMT interface (2 wait states) is shown. Note that the TB also can insert wait states during the writing of the request register (not shown in the diagram!).

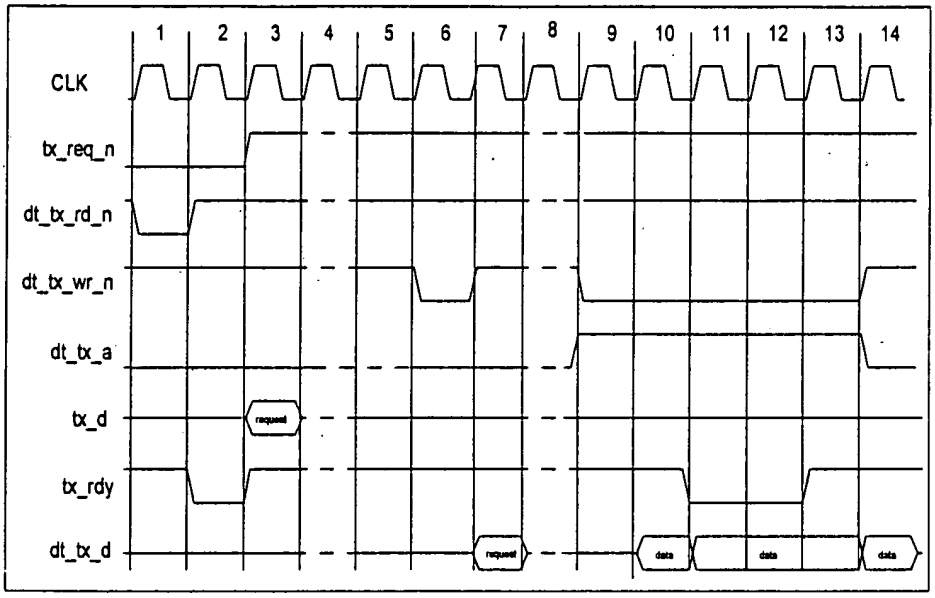


Figure 3
Complete transfer cycle

The efficiency of the data transfers improves with burst length. The peak throughput (not sustainable) can approach 1 word/clock with long bursts (theoreticly; M256F 1 word/3 clocks). However it is ultimately limited by the PMT interface access of the internal transmit buffer RAM. Each word the protocoll machine requests during a TB-DMUT data cycle may results in 2 additional wait states on tx_rdy.

The amount of transferred data depends on the requested amount of data by the TB and the open requests for this channel. The DMUT checks whether the current data section contains enough data. If not the DMUT will first transfer the rest of the current data section (write request register with STL = rest of current data section, transfer data). If the FE bit is not set in the Current Transmit Descriptor the DMUT will branch to the next Descriptor and transfer the rest of requested transfer (write request register, transfer data). **Figure 4** illustrates the 2 step transmission. Note that between writing the request register and the data transfer itself a couple of clock cycles may elapse since the DMUTFPI requests the data out of the shared memory via the FPI initiator bus.

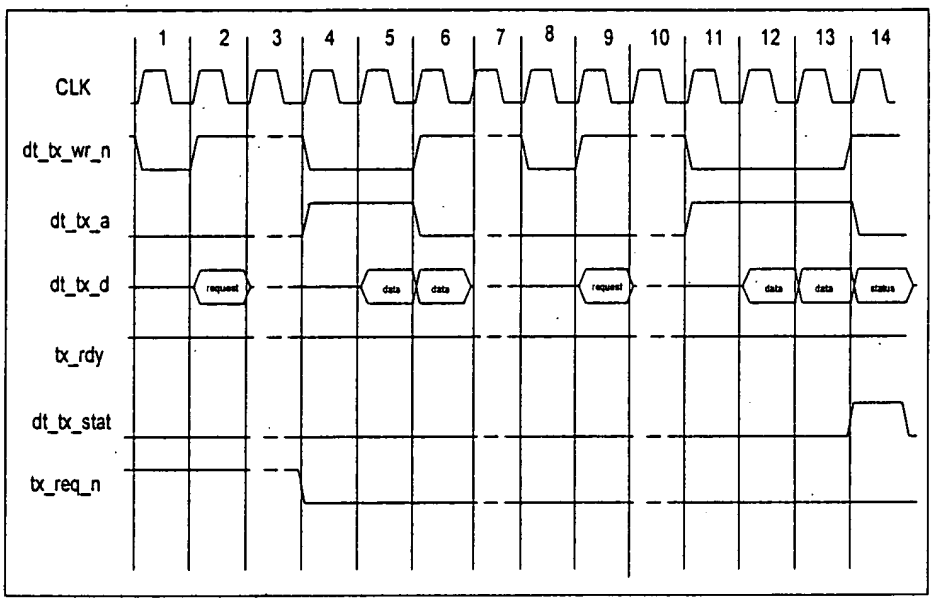


Figure 4
Transfer to TB from different datasection of TX Descriptor

For the last word of a frame transferred to TB or if a transmit abort occurred because HOLD=1 and FE=0 or NO=0 and FE=0 are detected in the transmit descriptor or a transmit abort is performed in the middle of a frame via the command register, the DMUT transfers a status word and activates the dt_tx_stat line (at the beginning of the cycle during writing the request register the CI bit will be set - status word indication). In this case the word has the following format:

Table 7
Status word

Bit	31	30	29..26	25	24	23 16	15 . . . 8	7 0
Function	FE	TAB	Reserved	BE1	BE0	Byte2	Byte1	Byte 0

BE[0..1]=00: Byte 0-2 are invalid, the word contains only status information

BE[0..1]=10: Byte 0 is valid, Byte 1-2 are invalid user data.

BE[0..1]=01: Byte 0-1 are valid, Byte 2 is invalid user data.

BE[0..1]=11: Byte 0-2 are valid

TAB (Transmit Abort) allows the protocol machine to send an abort sequence for the frame. FE allows the protocol machine to close the transmission of a frame (e.g. with CRC and flag).

3.2.4 DMUI Interface

The DMUT transfers the interrupt vectors to the DMUI (Interrupt controller) which will arbitrate the FPI initiator bus and transfer the vector in the corresponding interrupt queue in the shared memory. The interface between DMUT and DMUI is a bus shared by all the blocks generating interrupts. Therefore the transfer starts with an arbitration cycle and it might take some cycles until the request is granted, minimum latency is 1 clock cycle. Once the bus is granted to DMUT, DMUT deactivates its request in the next cycle and writes the vector on the dt_int_d data bus in the same cycle (because no address cycle is needed, additionally dt_int_d_en will be asserted). Since the DMUI will not insert wait states, a RDY signal from the DMUI is not necessary.

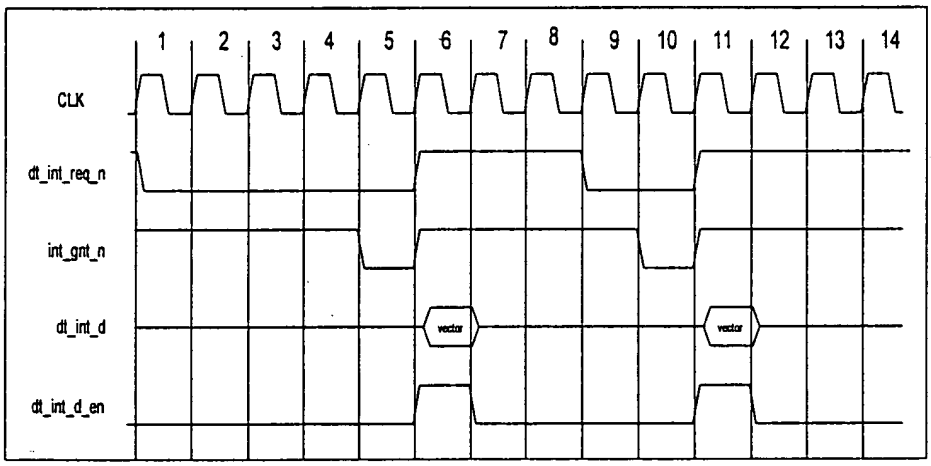


Figure 5
Interrupt Vector Transfer to the DMAI

3.3 Macro Functional Test

3.4 Macro Production Test

4 Register Description

4.1 Register Overview

The DMUT macro will be used in a system. Within the system it will happen that certain configuration information will be used by more than one macro. System programming would be very ineffective if these configuration information will be stored at different location in different macros. That's why a broadcast programming will be introduced. In order to set up channels a virtual channel specification, consisting of a set of registers, selected by the signal tfpi_vc_sel_n, will be programmed. Global information will be stored in a set of registers selected by the signal tfpi_vg_sel_n. Every macro implements only the registers or parts of the registers which contain necessary information.

All registers will be accessed via the target FPI bus, inside the macro the SMIF-BPI interface will control these accesses.

Essential for all slave register are fast accesses in write direction (i.e. no PCI wait states) and that a mechanism is provided to ensure that accesses are completed internally before the next command is given (i.e. no erroneous overwrite of data).

Read accesses are non critical, but a debug read back of all registers (virtual and standard) must be ensured;

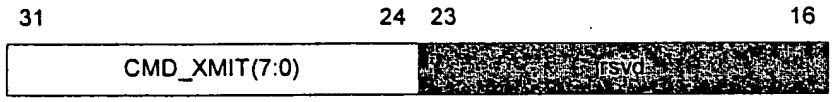
Test mode accesses are non critical and can deliver deliberate results; i.e. read back RAM contents

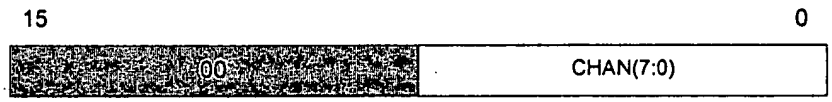
The following chapter describes a subset of registers, which are necessary for programming the DMUT macro. The reset values are the driven values from the DMUT, in the system environment they might be different (non implemented bits by the DMUT might vary)

4.2 Detailed Register Description of the Virtual Channel Specification (selected by tfpi_vc_sel_n)

4.2.1 Virtual Channel Specification Command (CSPEC_CMD)

Access	: read/write
Address	: 00_H
Reset Value	: 00000000_H





The reserved bits are implemented by others macros within the system.

The following commands will be supported:

Transmit Init
This command will cause the initialization of the channel CHAN(7:0). The DMUT will store First Transmit Descriptor Address (CSPEC_FTDA register) in its on-chip RAM. With the next request for this channel from the transmit buffer the entire descriptor pointed by the CSPEC_FTDA will be retrieved and afterwards a Command Complete Interrupt (CMDC) will be generated.

Transmit Off
The DMUT does not interprete this command, it will be done by the transmit buffer and the transmit buffer sets the DEL bit. The DMUT macro will see the DEL bit set for this channel in the request register of the transmit buffer. After reading this request register containing the DEL bit all open requests for this channel will be removed and and a Command Complete Interrupt (CMDC) will be generated. The C-bit will be set in the old Transmit descriptor and a HI INTR generated, if necessary (CEN, HI).

Transmit Abort
If this command is detected the First Transmit Decriptor Address (CSPEC_FTDA Register) will be stored. With the next request for this channel the current transmit descriptor will be suspended.

If the frame was correctly closed (FE), the new transmit descriptor, pointed by FTDA, will be retrieved.

If a frame was not sent completely (no FE) the DMUT will send a transmit abort status word to the transmit buffer and generate a transmit abort interrupt (for transparent mode these 2 actions will always be executed). The C-bit will be set in the old Transmit descriptor and a HI INTR issued, if necessary (CEN, HI).) Then the new transmit descriptor, pointed by FTDA, will be retrieved.

As soon as the entire descriptor is stored in the onchip memory, a Command Complete Interrupt will be generated.

When the DMUT has seen the HOLD bit for a channel (this channel is on hold), also an Transmit Abort Command can be issued, with the next request for the channel the DMUT

will branch immediately to the transmit descriptor pointed by the CSPEC_FTDA without repolling the old one

Transmit Hold Reset

Always when the CPU removes the HOLD bit in the transmit descriptor chain of the channel additionally it has to write this Transmit Hold Reset Command (THR). Internal action refer to description of the polling mechanism (chapter 2.2). When the DMUT repolls the old transmit descriptor and the hold bit is still set (erronoeus programming of the software), no special action takes place.

Transmit Debug

All register belonging to the Virtual Channel Spec will be read with this command. The DMUT provides the contents of all the registers (CSPEC_FTDA only valid after the first request for the channel, contains the current transmit descriptor) described in this section.

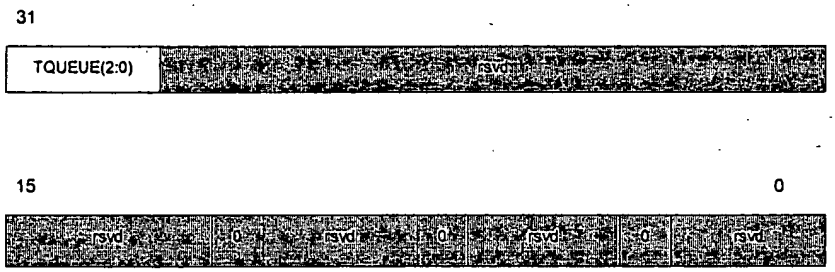
Issuing the Transmit Hold Reset (THR) Command will be accepted immediately without acknowledgement. After executing one of the other commands (init, off, abort) by the DMUT macro a Command Complete (CMDC) will be generated. The CPU may not write another command for this particular channel into the Command Register before the CMDC INTR occured. The commands are coded in the following way (the commands transmit idle and transmit update do not have an impact for the DMUT, they will be used by other macros in the system:

Command Table Receive:

31	30	29	28	27	26	25	24	
Rsvd	Update	Idle	Debug	HOLD RESET	ABORT	OFF	INIT	function
0	0	0	0	0	0	0	1	transmit init
0	0	0	0	0	0	1	0	transmit off
0	0	0	0	0	1	0	0	transmit abort
0	0	0	0	1	0	0	0	transmit hold reset
0	0	0	1	0	0	0	0	transmit debug
0	0	0	0	0	0	0	0	transmit nop
0	0	1	0	0	0	0	0	transmit idle
0	1	0	0	0	0	0	0	transmit update

4.2.2 (Virtual) Channel Specification Buffer (CSPEC_BUFFER)

Access	: read/write
Address	: 20_H
Reset Value	: 00000000_H

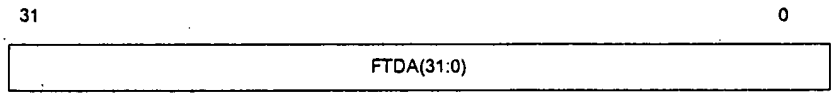


Note: The reserved bits are implemented by other macros within the system.

TQUEUE: The generated transmit channel interrupts will be sent to this interrupt queue (command INTR will be sent to the command queue)

4.2.3 (Virtual) Channel Specification FTDA (CSPEC_FTDA)

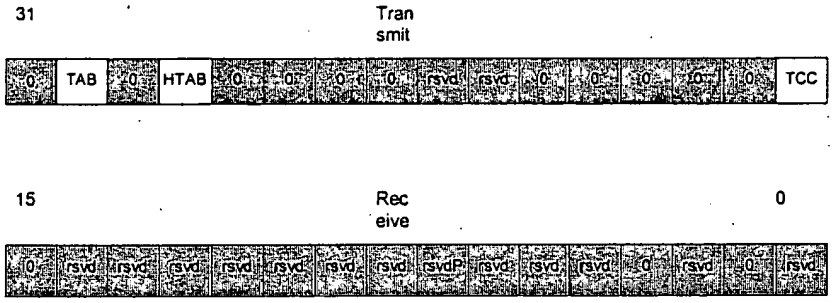
Access : read/write
Address : 28_H
Reset Value : 00000000_H



FTDA(31:0): First Transmit Descriptor Address (dword aligned)

4.2.4 (Virtual) Channel Specification Interrupt Mask/Queue (CSPEC_IMASK)

Access : read/write
Address : $2C_H$
Reset Value : 00000000_H



Note: The reserved bits are implemented by other macros within the system.

Interrupt Generation Mask: These bits correspond to the respective channel interrupts, if set to '1', the corresponding interrupt will not be generated by the macro

4.3 Detailed Register Description of the Global registers (selected by tfpi_vg_sel_n)

4.3.1 Configuration Register 1 (CONF1)

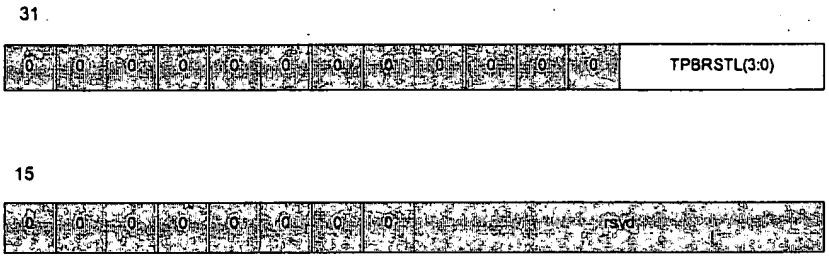
Access : read/write
Offset Address : 40_H
Reset Value : 00000000_H

Note: The reserved bits are implemented by others macros within the system.

LBE: Little/Big Endian Byte Swap: '0': Little Endian ; '1' : Big Endian

4.3.2 Configuration Register 3(CONF3)

Access : read/write
Offset Address : 48_H
Reset Value : 00090000_H



Note: The reserved bits are implemented by others macros within the system.

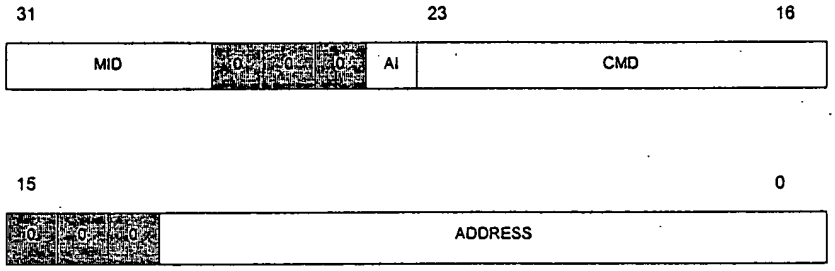
TPBRSTL(3:0): Transmit packet burst length
Sets the maximum PCI burst length for tramsit packet data

0000	1 DWORD
0001	4 DWORDs
0010	8 DWORDs
0011	12 DWORDs
0100	16 DWORDs
0101	24 DWORDs
0110	32 DWORDs

0111	40 DWORDs
1000	48 DWORDs
1001	64 DWORDs

4.3.3 Test Command Register (TAC)

Access : write/read
Offset Address : 58_H
Reset Value : 00000000_H



MID: Macro ID Code (Bit 31:28 = 0110 for DMUT)
AI: Auto Increment Function
Address: internal address
CMD: Command (select of ram and register)

CMD:

23	22	21	20	19	18	17	16	function
0	0	0	0	0	0	0	1	RAM_ID
0	0	0	0	0	0	1	0	RAM_NXPTR
0	0	0	0	0	0	1	1	RAM_OATPTR
0	0	0	0	0	1	0	0	RAM_CURPTR
0	0	0	0	0	1	0	1	RAM_NO
0	0	0	0	0	1	1	0	R_REG
0	0	0	0	0	1	1	1	ADDR_REG
0	0	0	0	1	0	0	0	CHN_REG

23	22	21	20	19	18	17	16	
0	0	0	0	1	0	0	1	RTL_REG
0	0	0	0	1	0	1	0	REQ_REG
0	0	0	0	1	0	1	1	FPID_REG
0	0	0	0	1	1	0	0	TC_REG
0	0	0	0	1	1	0	1	FLAG_REG
0	0	0	0	1	1	1	0	REM_REG
0	0	0	0	1	1	1	1	NO_REG

The Address in the Command Register is only for RAMx accesses (command 1 - 5) valid, address width is 8 bit. Autoincrement will only be supported for RAM Access.

General

- the test access provides read/write access of important internal rams and registers
- test registers are virtuals global registers (SEL signal: pb_vg_tfpi_sel_n / implemented as a daisy chain).
- the single macros are selected by the MID code of the test command register
- CMD specifies/selects one of the macro rams/registers
- the address field is used to access a ram address
- AI: autoincrement : address given in the address field is incremented automatically for each access
- All macros which are not selected by MID drive data output "00000000" and <macro>_TPFI_RDY = '1'. Driving "00000000" would mean not disable the enable line for data out, but to set the output data to "00000000".

Test write access:

1. Write TAC
2. Write TAD

Test read access:

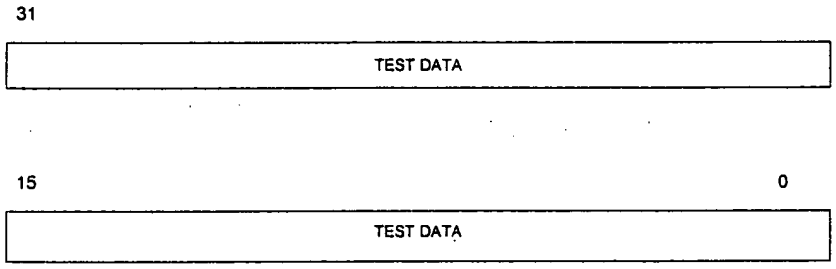
1. Write TAC
2. Read TAD

Typically the macro selected via MID delays the RDY signal until the selected ram/ register has been read and the data can be provided at the TFPI interface. No prefetch of testdata is required.

Note: there is no read/write selection in the CMD field; rd/wr's are handled with the TFPI read & write signals

4.3.4 Test Data Register (TD)

Access	: read/write
Address	: $5C_H$
Reset Value	: 00000000_H



TEST DATA: ram/register test data (read or write)

Note: It is assumed that a TFPI_SEL_N signal is generated for register access of common TAC and TD register. For write / read access the MID (macro ID) is used for addressing one macro: If MID does not match with the hard coded ID, TFPI_D are prepared corresponding the 'daisy chain' requirements.I.e. each macro drives "00000000". These '0' s are ored in the macroshell.

5 Functional Test Specification (Macrolevel)

(use Checklist C06 "Function Checklist - Comparison of Specification vs. Circuit"
http://www.hl.siemens.de/lognet/hl_ta/dhb/f.htm)